|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

##### ФАКУЛЬТЕТ ИУ «Информатика и системы управления»

**КАФЕДРА ИУ8 «Компьютерная безопасность*»***

**ЛАБОРАТОРНАЯ РАБОТА №4-5**

**Название лабораторной работы: Использование объектов своих классов в последовательных контейнерах библиотеки STL.**

**по курсу: Алгоритмические языки.**

**Группа ИУ8 – 24**

**Студент: Полонский Сергей**

**Преподаватель: Барыкин Дмитрий**

**Дата: 07.04.2024**

**ОТЧЕТ:**

**1)** **Цель работы:** научиться использовать контейнеры, которые содержат объекты своих классов.

**2)** **УСЛОВИЕ ЗАДАЧИ:** Объект- банковский кредит (поля: название, сумма кредита, тип валюты, ставка в % годовых). Выполнить сортировку элементов контейнера по сумме кредита. Исходный контейнер deque, копируем в list.

**3) Текст программы с коментариями:**

#include <iostream>

#include <fstream>

#include <string>

#include <deque>

#include <list>

#include <algorithm>

#include <set>

#include <unordered\_set>

using namespace std;

class Kredit { // Класс кредит

friend void operator <<(ofstream& t, Kredit& b) { // оператор вывода полей класса в поток

t << "name:" << b.name << " " << "summa:" << b.sum << " " << "valuta:" << b.type\_v << " " << "percent:" << b.percent << endl;

}

public:

Kredit(); // конструктор по умолчанию

Kredit(string a, double b, string c, double d); // конструктор со значениями

Kredit(const Kredit& a); // конструктор копирования

Kredit(Kredit&& a); // конструктор перемещения

void operator =(const Kredit& b); // оператор = копирования

void operator =(const Kredit&& b); // оператор = перемещения

bool operator <(const Kredit& b) const; // оператор < сравнения классов

string getname() const{ return name; } // геттеры полей, будут использованы для получения хэша

double getsum() const{ return sum; }

string gettype\_v() const{ return type\_v; }

double getpercent() const{ return percent; }

bool operator ==(const Kredit& b) const { // оператор "равно" для unordered\_set

return sum == b.sum;

}

private:

string name; // имя кредита

double sum; // сумма кредита

string type\_v; // тип валюты

double percent; // проценты кредита

};

struct KreditHesh { // функтор для создания хэша

hash<string> shash;

hash<double> dhash;

size\_t operator()(const Kredit& a) const {

const size\_t coef = 5171;

return (coef \* coef \* coef \* shash(a.getname()) + coef \* coef \* dhash(a.getsum()) + coef \* shash(a.gettype\_v()) + dhash(a.getpercent()));

}

};

Kredit::Kredit() : name("noname"), sum(0), type\_v("unknown"), percent(0) {}

Kredit::Kredit(string a, double b, string c, double d) {

name = a;

sum = b;

type\_v = c;

percent = d;

}

Kredit::Kredit(const Kredit& a) {

name = a.name;

sum = a.sum;

type\_v = a.type\_v;

percent = a.percent;

}

Kredit::Kredit(Kredit&& a) {

swap(name, a.name);

swap(sum, a.sum);

swap(type\_v, a.type\_v);

swap(percent, a.percent);

}

void Kredit::operator =(const Kredit& b) {

name = b.name;

sum = b.sum;

type\_v = b.type\_v;

percent = b.percent;

}

void Kredit::operator =(const Kredit&& b) {

name = b.name;

sum = b.sum;

type\_v = b.type\_v;

percent = b.percent;

}

bool Kredit::operator <(const Kredit& b) const{

return sum < b.sum;

}

int main()

{

ifstream file("input.txt"); // связывание файла с файловым потоком

string l1, l2, l3, l4, p; // переменные для считывания информации из файла

getline(file, p);

int n = atoi(p.c\_str()); // количество переменных класса(число в начале файла)

deque<Kredit> deq1;

while (!file.eof()) {

getline(file, l1, '/');

getline(file, l2, '/');

getline(file, l3, '/');

getline(file, l4, '/');

getline(file, p); // в файле каждая строка оканчивается служебным сиволом /n, этот гитлайн нужен для того чтобы его съесть

deq1.push\_back(Kredit(l1, atof(l2.c\_str()), l3, atof(l4.c\_str())));

}

list<Kredit> list1;

sort(deq1.begin(), deq1.end()); // сортировка deque с помощью перегрузки оператора <

//cout << deq1[1];

copy(deq1.begin(), deq1.end(), back\_inserter(list1));

//cout << deq1[0];

//cout << list1.front();

ofstream gogo("output.txt"); // связывание потока вывода с файлом

if (gogo.is\_open()) for (auto& i : list1) gogo << i;

gogo.close(); // закрытие файла

file.close(); // закрытие файла

set<Kredit> aboba;

aboba.insert(Kredit("Losos", 11, "dollar", 1));

unordered\_set<Kredit, KreditHesh> boaba;

boaba.insert(Kredit("Paltus", 12, "Shekel", 228));

}}